Software-Projekt I

Prof. Dr. Rainer Koschke

Arbeitsgruppe Softwaretechnik
Fachbereich Mathematik und Informatik
Universität Bremen

Sommersemester 2013

Allgemeines zur Softwaretechnik I

Fragen

- Was ist eigentlich Software?
- Worin unterscheidet sich Software von anderen menschlichen Erzeugnissen?

Was ist Software?

Definition

Software: Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.

IEEE Std 610.12-1990 (1990)

Software umfasst also nach IEEE Std 610.12 Programme, Abläufe, Regeln, auch Dokumentation und Daten, die mit dem Betrieb eines Rechnersystems zu tun haben.

Software

- ist ein technisches Produkt,
- weist aber einige einmalige Merkmale auf.

$\rightarrow$ Sie sollten beim Umgang mit Software beachtet werden.

Generell kann und sollte Software als technisches Produkt betrachtet werden, das wie andere Produkte von Ingenieuren systematisch entwickelt werden kann und durch feststellbare Eigenschaften (Funktionalität, Qualität) gekennzeichnet ist. Software ist sogar frei von allen Unwägbarkeiten, die anderen technischen Produkten anhaften. Sie ist in diesem Sinne das ideale technische Produkt. Als Software-Leute sollten wir darum keine spezielle Nachsicht erwarten. Software weist aber (wie viele andere Produkttypen) einige besondere und wenigstens in dieser Kombination einmalige Merkmale auf: Sie müssen beim Umgang mit Software beachtet werden.

Eigenschaften von Software

Software ist immateriell

- Software kann nicht ohne Weiteres betrachtet werden.
- Kopie und Original sind völlig gleich.
- Software wird nicht gefertigt, sondern nur entwickelt.
- Software verschleißt nicht.
- Software „altert“ in dem Maße, in dem sich ihre Umwelt ändert.

Was wir als natürlich empfinden, ist an materielle Eigenschaften geknüpft. An Software ist nichts natürlich. Erfahrungen aus der natürlichen Welt sind nicht übertragbar (werden dennoch ständig übertragen, siehe z.B. Wartung). Kopie und Original sind völlig gleich. Software wird nicht gefertigt, sondern nur entwickelt. Software verschleißt nicht; die Wartung stellt nicht den alten Zustand wieder her, sondern einen neuen. Wiederverwendung ist bei Software extrem lukrativ, wenn der Aufwand für Anpassungen relativ gering ist. Programmierer unterschätzen meist das Problem (. . . oder überschätzen sich selbst!)

Eigenschaften von Software

Programme verhalten sich oft unstetig.

```
public int unstetig(int x)
{
  if (x == 42)
    return 0;
  else
    return 2 + x/2;
}
```


Ein Programm realisiert nicht unbedingt eine stetige Funktion. Der Zusammenhang zwischen Eingabe und Ausgabe lässt sich nicht durch eine kontinuierliche Funktion beschreiben.

Das hat Konsequenzen zum Beispiel für den Test. Wenn eine Funktion für die Eingabe 1 und für die Eingabe 100 funktioniert, kann man daraus nichts über ihr Verhalten für die Wert 2...99 schließen.

Eigenschaften von Software

Korrektheit ist durch Test nicht prüfbar.

Eigenschaften von Software

- Software hat keine natürliche Lokalität.
- Strukturen müssen aktiv geschaffen werden.

Natürliche Lokalität gibt es bei Software nicht. Im Speicher eines Rechners gibt es keine schützende Distanz.

Eigenschaften von Software

Ihre Werkstoffe sind amorph und universell.

Die Werkstoffe der Software sind amorph und universell. Werkstoffe sind die eingesetzten Sprachen, meist nur die natürliche Sprache und die Programmiersprache, evtl. noch andere, z.B. graphische Notationen. Damit lassen sich Datenbanken, Tabellenkalkulationsprogramme, Textverarbeitungssystem, Spiele und alles andere programmieren.

Eigenschaften von Software

Software-Systeme sind sehr komplex:

→ Entwicklungskosten sind unvermeidlich hoch.

→ Korrekte Konstruktion ist extrem schwierig.

Software-Systeme sind sehr komplex (nach Zahl der relevanten Entscheidungen darin), vergleichbar mit anderen sehr komplexen Systemen. Es ist extrem schwer, sie trotzdem so zu konstruieren, dass sie automatisch und korrekt arbeiten.

Eigenschaften von Software

Software spiegelt (in vielen Fällen) die Realität wider.

> *"This complexity is compounded by the necessity to conform to an external environment that is arbitrary, unadaptable, and everchanging."*
>
> –F.P. Brooks, 1987

Eigenschaften von Software

Software gilt als flexibel (d.h. leicht änderbar)

- Sie verbindet Hardware und Umgebung.
- Details der Aufgabenstellung werden in aller Regel durch Software realisiert.

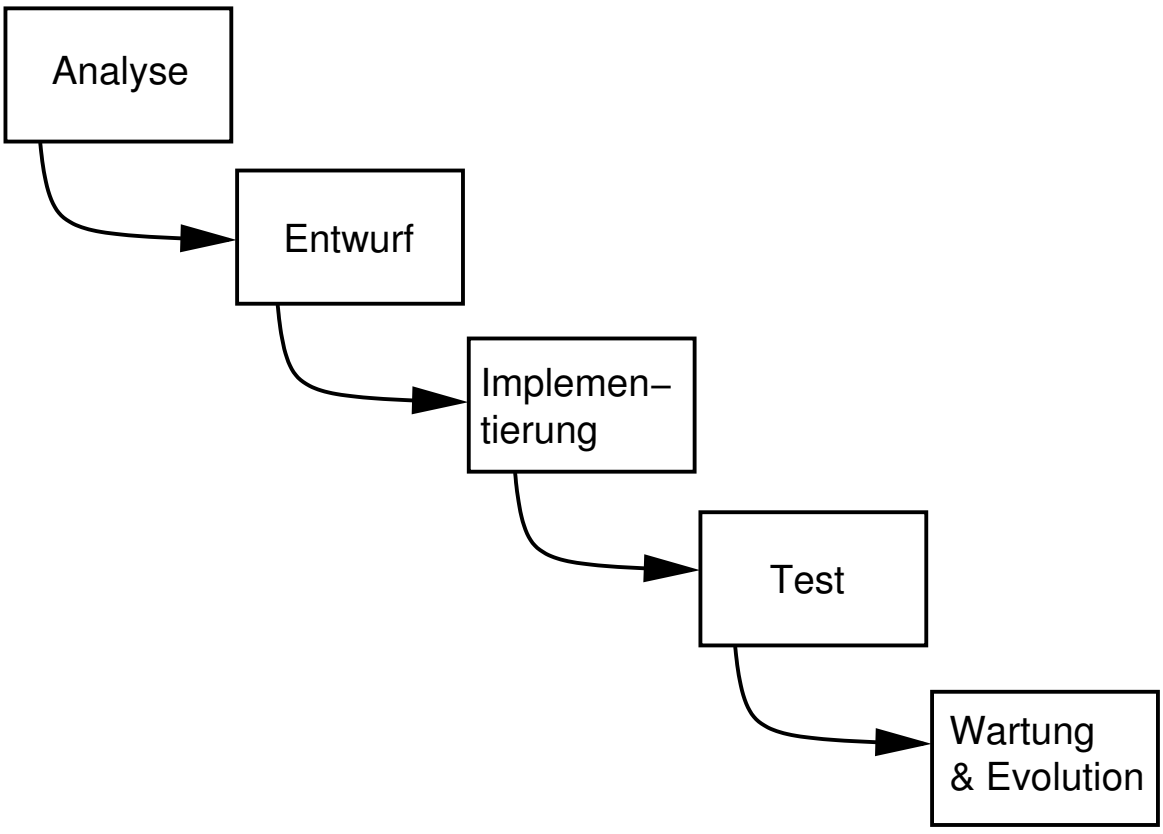
$\rightarrow$ Änderungen schlagen sich entsprechend primär in der Software nieder.

Software ist nur in dem Maße flexibel, in dem die Flexibilität explizit eingebaut ist: Es ist leicht einen Parameter zu verändern, aber nur wenn dieser Parameter in der Software vorhanden ist.

Fragen

- Was sind die wesentlichen Phasen der Softwareentwicklung?
- Wie lange dauern diese Phasen?
- Und wann ist Software fertig entwickelt?

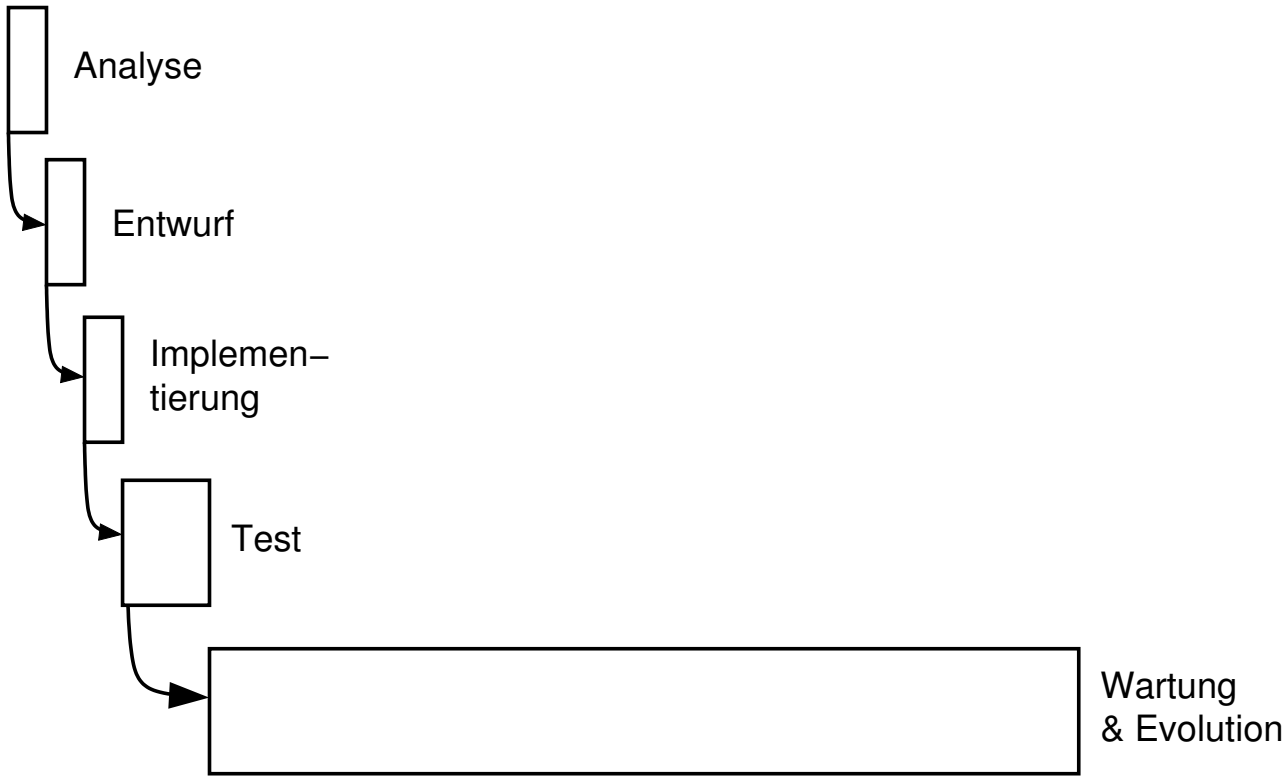
Software-Lebenszyklus



Wie jeder komplexe Engineering-Prozess, so verläuft auch die Software-Entwicklung in mehreren Phasen. Die Software wird spezifiziert, eine Lösung wird entworfen und implementiert und schließlich getestet. Nach der Auslieferung wird sie gewartet. Das heißt, Fehler werden korrigiert sowie Erweiterungen und Anpassungen vorgenommen.

Diese Darstellung suggeriert, dass alle Teilphasen mehr oder minder gleich lang sind. Dies stimmt jedoch nicht.

Aufwand in den Phasen nach McKee (1984)



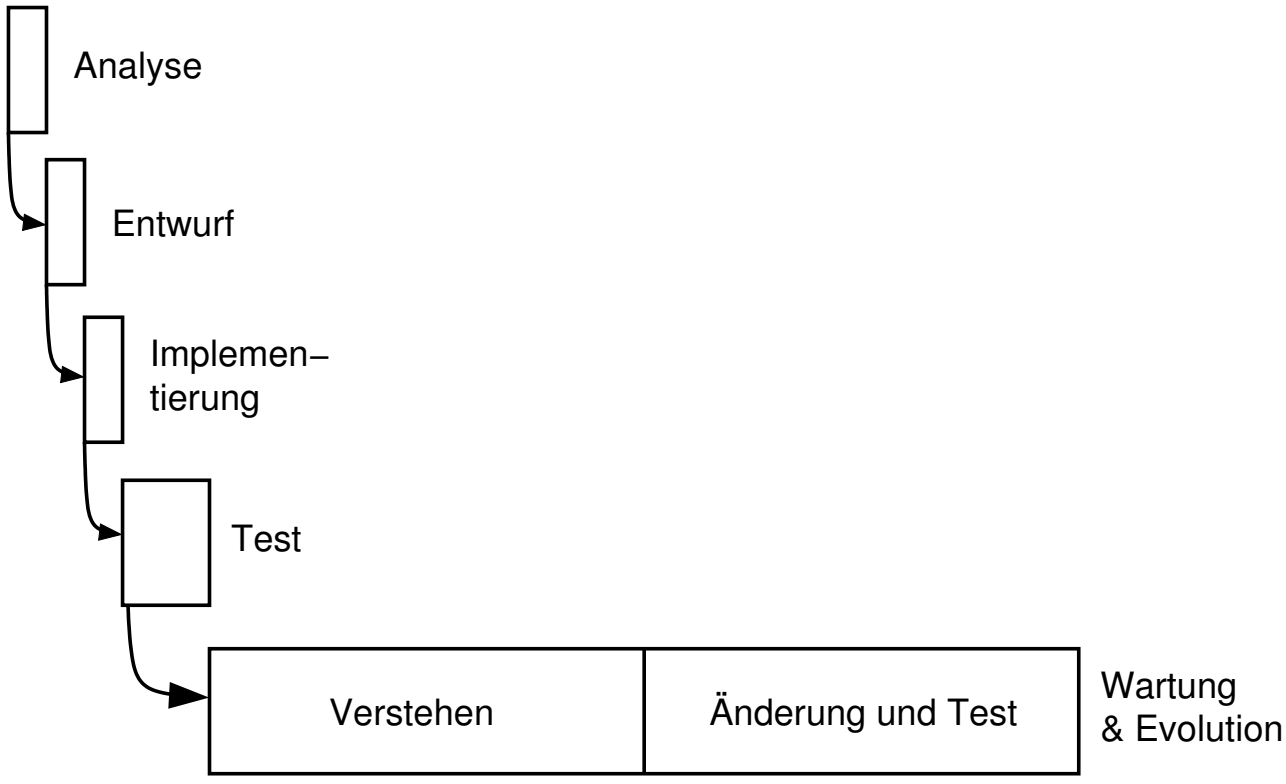
Hier seht Ihr eine Darstellung des Entwicklungsprozesses, in der die einzelnen Phasen entsprechend ihrer Dauer gezeichnet sind.

Empirische Untersuchungen zeigen, dass bis zu 80% der Kosten erst nach der Auslieferung entstehen, d.h. in der Wartungs- und Evolutionsphase.

Diese Phase unterscheidet sich von den anderen Phasen darin, dass man sich mit einem existierenden System auseinander setzen muss.

Diese Verteilung sagt übrigens auch aus, dass unsere Studenten mit einer Wahrscheinlichkeit von 80 Prozent in ihrem späteren Berufsleben mit der Wartung bestehender Systeme zu tun haben werden. Nur an wenigen Universitäten werden sie hierauf ausreichend vorbereitet.

Evolutionsphase nach Fjeldstadt und Hamlen (1984)



Weitere Studien zeigen, dass Wartungsprogrammierer etwa die Hälfte ihrer Zeit allein mit der Analyse der Software verbringen, ehe sie ihre Änderungen vornehmen und testen. Bei der Korrektur von Fehlern liegt der Aufwand eher bei 60%.

Dieser hohe Aufwand liegt an der unzureichenden Information, die den Wartungsprogrammierern zur Verfügung steht. Die Programmierer müssen die Informationen mühsam von Hand herleiten, weil ihnen die dazu notwendigen Werkzeuge fehlen.

Wer Kosten bei der Software-Entwicklung sparen möchte, muss hier ansetzen.

Fragen

- Warum muss Software angepasst werden und was passiert dabei?

Lehman und Beladys „Gesetze“ (1985)

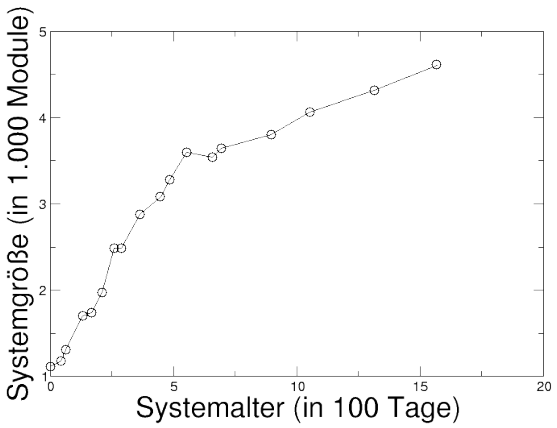
Gesetz vom fortwährenden Wandel

- Software löst ein Problem der realen Welt,
- die reale Welt ändert sich,
- Software muss sich anpassen,
- bis sie abgelöst wird.

Lehman und Beladys „Gesetze“ (1985)

Gesetz der zunehmenden Komplexität

- die Komplexität der Software erhöht sich beim Wandel,
- wenn keine Gegenmaßnahmen ergriffen werden



Leider geht mit dem Zwang zur Änderung der Anstieg der Komplexität der Software einher. Hier ist eine Grafik von Many Lehman zu sehen, der sich als einer der ersten mit dem Phänomen der Software-Evolution empirisch auseinander gesetzt hat.

Es zeigt den Umfang eines Systems, gemessen in der Anzahl seiner Module, und das Wachstum dieses Umfangs über die Zeit. Die Messpunkte sind die verschiedenen Releases der Software.

Das Wachstum selbst ist deutlich sichtbar. Aber gleichzeitig ist eine Verlangsamung des Wachstums erkennbar. Ebenso nimmt die Dauer zwischen Releases zu.

Um diesem Trend entgegen zu wirken, müssen Maßnahmen ergriffen werden, um die Komplexität einzudämmen.

Software-Evolution

Konsequenzen:

- Änderbarkeit ist eine Schlüsselqualität
- Änderbarkeit muss frühzeitig angestrebt werden
  - sie lässt sich nicht nachträglich überstülpen
- wer sie vernachlässigt, nimmt einen Kredit auf, den er teuer abzahlen wird

Fragen

- Woher kommt das Software-Engineering (die Softwaretechnik)?

Wo liegen die Probleme in der Softwaretechnik?

"The whole trouble comes from the fact that there is so much tinkering with software. It is not made in a clean fabrication process, which it should be. What we need is software engineering."

–F.L. Bauer, 1968

In den späten 60er Jahren:

- Manche Software-Projekte sind selbst mit gigantischem Aufwand nicht zu schaffen.
- In vielen anderen Fällen wird der Abschluss nur mit unbefriedigenden Resultaten, viel zu spät und/oder mit extremen Kostenüberschreitungen erreicht.
- Andererseits erzielt die Hardware-Entwicklung atemberaubende Fortschritte.

Kurz: Es ist nicht alles machbar, was prinzipiell möglich sein müsste. Für diese Probleme kam das Schlagwort Software Crisis in Mode. Die Software-Krise war Anlass für viele Diskussionen und Tagungen. Auf der NATO-Konferenz in Garmisch (1968) prägte F.L. Bauer das Wort Software Engineering als Provokation (siehe dazu Bauer, 1993).

Wie viel hat sich bis heute daran geändert?

Beispiel

- Fehler:

  vollständiger Batterieausfall in Fahrzeugen eines deutschen Automobil-Herstellers
- Ursache:

  Fehler in der software-gesteuerten Innenbeleuchtung
- Folge:

  Rückrufaktion mit einem (geschätzten) Schaden von mehreren Millionen DEM

Partsch (1998)

Wie viel hat sich bis heute daran geändert?

Beispiel

- Fehler:

  Bei der rechnergestützten Auswertung der OB-Wahl in Neu-Ulm 1994 wurde zunächst eine Wahlbeteiligung von 104% festgestellt.
- Ursache:

  In der Auswertungssoftware hatte sich ein mysteriöser Faktor 2 eingeschlichen.
- Folge:

  Neuauszählung

Partsch (1998)

Wie viel hat sich bis heute daran geändert?

Beispiel

- Fehler:

  USS Yorktown treibt während eines Manövers im September 1997 manövrierunfähig bei Cape Charles, VA.
- Ursache:

  Fehler in einer Routine zum Abfangen einer Division durch Null in einer Windows NT Applikation. Die „Null“ wurde als manuelle Eingabe einer enormen Datenmenge interpretiert.
- Folge:

  “Atlantic Fleet officials said the ship was dead in water for about 2 hours and 45 minutes.”

Neumann (1999)

Produktgarantie?

Disclaimer of Warranty:

> This **car** is provided under this license on an “as is” basis, **without warranty of any kind**, either expressed or implied, including, **without limitation, warranties that the car is free of defects, merchantable, fit for a particular purpose or non-infringing**. The **entire risk** as to the quality and performance of the **car is with you**. Should any **car** prove defective in any respect, you (not the initial developer or any other contributor) assume the cost of any necessary servicing, repair or correction.

Garantien über Software?

Disclaimer of Warranty:

> **This software** is provided under this license on an “as is” basis, **without warranty of any kind**, either expressed or implied, including, **without limitation, warranties that the software is free of defects, merchantable, fit for a particular purpose or non-infringing**. The **entire risk** as to the quality and performance of the **software is with you**. Should any **software** prove defective in any respect, you (not the initial developer or any other contributor) assume the cost of any necessary servicing, repair or correction.

An solche Disclaimer haben wir uns schon lange gewöhnt. Stellt euch vor, ihr lest einen solchen in der Betriebsanleitung eures Autos. Ihr würdet das Auto sofort zurückgeben.

Übrigens: Solche Disclaimer sind nach deutschem Recht hinfällig. Stellt ihr jemandem im Netz ein Programm zur Verfügung, seid ihr automatisch in der Haftung. Grobe Fahrlässigkeit und Vorsatz lassen sich in Deutschland nicht ausschließen.

Wenn der Zweck eures Programms nicht spezifiziert ist, so wird zu Grunde gelegt, was man üblicherweise von einem Programm dieser Art erwartet.

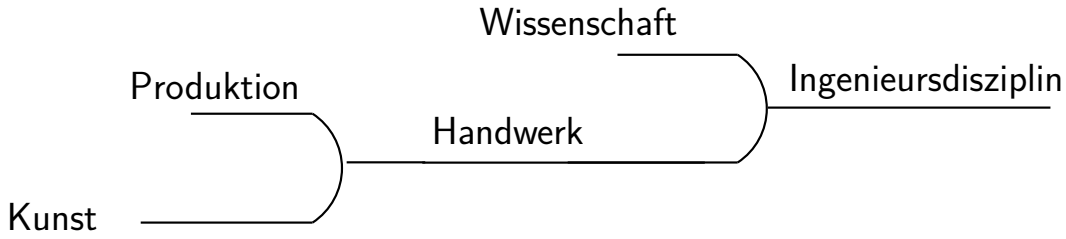
Fragen

- Was ist mit dem Begriff *Engineering* in *Software-Engineering* verbunden?

Ingenieursdisziplin (Shaw und Garlan 1996)

Geübte Arbeiter

- etabliertes Vorgehen
- pragmatische Verfeinerungen
- Ausbildung in der Ausführung
- ökonomische Betrachtung der Materialien
- Herstellung für den Verkauf



Talentierte Amateure

- intuitiv und ad hoc
- wahlloser Fortschritt
- gelegentliche Übermittlung
- verschwenderische Verwendung verfügbaren Materials
- Herstellung für die Verwendung, weniger für den Verkauf

Ausgebildete Personen

- Analyse und Theorie
- Fortschritt basiert auf Wissenschaft
- Einführung neuer Anwendungen durch Analyse
- Marktsegmentierung durch Produktvielfalt

Definition

Softwaretechnik:

- Entdeckung und Anwendung solider Ingenieur-Prinzipien mit dem Ziel, auf wirtschaftliche Art Software zu bekommen, die zuverlässig ist und auf realen Rechnern läuft.

  –F.L. Bauer

- Herstellung und Anwendung einer Software, wobei mehrere Personen beteiligt sind oder mehrere Versionen entstehen.

  –D.L. Parnas

- (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1). IEEE Std 610.12-1990 (1990)

Software Engineering wurde oft definiert. Drei wichtige Definitionen sind hier angegeben.

Die IEEE-Definition ist problematisch, wenn nicht falsch, weil sie wertet und fordert, also ein Ideal beschreibt, und einen anderen undefinierten Begriff zu Grunde legt.

Merkmale der alten Ingenieurdisziplinen

- Kostendenken als Grundlage von Bewertungen
- praktischer Erfolg als einziger zulässiger Beweis
- Qualitätsbewusstsein als Denkprinzip
- Einführung und Beachtung von Normen (Schnittstellen, Verfahren, Begriffe)
- Denken in Baugruppen

Definition und Anwendung von Methoden, Bereitstellung und Verwendung von Werkzeugen sind allgemein bewährte Ansätze, also nicht typisch für Ingenieure. Ingenieur-spezifisch sind dagegen:

- Kostendenken als Grundlage von Bewertungen (Achtung, es geht hier um die Gesamtkosten, nicht nur um Geld!)
- praktischer Erfolg als einziger zulässiger Beweis: nur was gezählt oder gemessen wurde, ist ein akzeptables Argument;
- Qualitätsbewusstsein: hohe Qualität ist ein Prinzip der unspezifischen Kostensenkung; kommt aus der handwerklichen Tradition, die die Ingenieure übernommen haben;
- Normen (vgl. Norm für Passstifte); Merke: Eine Norm wird immer beachtet!
- Baugruppen: Durch Normen entsteht die Möglichkeit, Baugruppen zu verwenden, also die Lösung der Teilprobleme auszuklammern.

Wiederholungsfragen

- Was sind die besonderen Eigenschaften von Software?
- Was folgt aus diesen Eigenschaften für die Softwareentwicklung?
- Aus welchen Phasen besteht der Software-Lebenszyklus?
- Was ist die Besonderheit der Wartungsphase und was folgt daraus für die Softwareentwicklung?
- Warum muss Software überhaupt angepasst werden? Welche Folgen können diese Änderungen haben?
- Welche Einsichten führten zur Entstehung der Softwaretechnik? Mit welchen Problemen ist die Softwaretechnik nach wie vor konfrontiert?
- Was ist Softwaretechnik? Welche Ziele verfolgt sie?
- Was sind die Kennzeichen einer Ingenieursdisziplin und wie entsteht sie?

1 Fjeldstadt und Hamlen 1984 Fjeldstadt ; Hamlen: Application Program Maintenance Study: Report to Our Respondents. In: Proc. GUIDE 48, IEEE Computer Society Press, April 1984

2 IEEE Std 610.12-1990 1990 : IEEE Standard Glossary of Software Engineering Terminology. IEEE Standard 610.12-1990. 1990

3 Lehman und Belady 1985 Lehman, M. ; Belady, L.: Program Evolution: Processes of Software Change. London : Academic Press, 1985

4 McKee 1984 McKee, J.R.: Maintenance as a Function of Design. In: AFIPS National Computer Conference, Las Vegas (Ch. 27), 1984

5 Neumann 1999 Neumann, Peter G.: Risks to the Public in Computers and Related Systems. In: ACM SIGSOFT Software Engineering Notes 24 (1999), Nr. 1, S. 31

6 Partsch 1998 Partsch, H.: Requirements-Engineering systematisch - Modellierung für softwaregestützte Systeme. Berlin, Heidelberg, New York : Springer-Verlag, 1998

7 Shaw und Garlan 1996 Shaw, Mary ; Garlan, David: Software Architecture – Perspectives on an Emerging Discipline. Upper Saddle River, NJ : Prentice Hall, 1996. – ISBN ISBN 0-13-182957-2